IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-2, Issue-9 E-ISSN: 2347-2693

Uses of HDFS in Metadata Management System

N.Durga[1*], D.Ragupathi[2] and V. Raj Kumar[3th]

[1]*Department of Computer Science, AVVM Sri Pushpam College, Bharathidasan University, India,*
[2] *Department of Computer Science, AVVM Sri Pushpam College, Bharathidasan University, India*
[3] *R&D of Computer Science, STEPINFOTECH, India*

www.ijcaonline.org

Received: 28 Feb 201X Revised: 10 March 201X Accepted: 22 March 201X Published: 30 March 201X

Abstract— A (HDFS) Hadoop Distributed File System is intended to store very large data sets unfailingly and to stream those data sets at elevated bandwidth to consumer applications. Metadata managing is serious to distributed file system. In HDFS design, a solo master server administers all metadata, while numerals of data servers pile up file data. This design can't gather the exponentially improved storage claim in cloud computing, as the solo master server may happen to a performance traffic jam. Proportional study of a metadata managing proposal is done. Its consist of three techniques tree separation, hashing and reliable hashing of metadata managing proposal. Out of these three proposals reliable hashing is the best techniques which utilize multiple Name Nodes, and segregate the metadata into "Buckets" which can be energetically move around among Name Nodes according to system job weight. To uphold consistency, metadata is simulated in dissimilar Name Nodes with log duplication knowledge, and Paxos algorithm is adopted to keep duplication uniformity.

Keywords— *Hadoop, HDFS, Distributed File System, File Management System, Metadata Management*

I. INTRODUCTION

With the brisk improvement of Internet, the quantity of data is on the swell, and the large size data storage and handing out has become a trouble. Cloud computing is one of the most fashionable explanation to meet the claim. Cloud computing provides shrink cost of hardware source and improved equipment employment. Users can contact all variety of Internet services and virtual computing influence through inconsequential convenient devices rather than long-established Pc. Cloud storage is an input subject for cloud computing, and metadata managing plays a serious position in Cloud. Metadata is the information that illustrates the arrangement of a file system, such as pecking order namespace, folders and file characteristic and record of file to data chunks. Even though the range of metadata is comparatively minute in the file system, metadata process can obtain up more than 50 percent of the complete file system process. So the metadata managing is extremely significant for file system Performance.

There are a range of distributed file systems like (HDFS) Hadoop Distributed File System, (GFS) Google File System, (PVFS) Parallel Virtual File System, and Lustre. The (GFS) Google File System and (HDFS) Hadoop Distributed File System are the most familiar file system set up in large scale distributed systems like Face book, Google and Yahoo in the present day.

Parallel Virtual File System is an open source RAID-0 technique parallel file system for clusters. It separation a file into stripe part and distributes these stripes to diskette in a round robin manner. PVFS will have one metadata server and numerous data servers. All data transfer of file contented flows among clients and data server nodes in parallel not including departing all the way through the metadata server. The lethal annoyance of PVFS is that it does not afford any fault-tolerance in its existing structure. The malfunction of every single server node will concern the entire file system.

Lustre is a collective diskette file system. Normally worn for large scale computing in Clusters. It's an open-standard support system with enormous modularity and compatibility with interrelate, networking mechanism and storage hardware. At this time, it is merely obtainable for the Linux Operating System.

(GFS) Google file system is an expandable distributed file system that chains the intense workload at the Google site and executes on a cluster with economical product hardware. In GFS, a solitary master node is used to uphold the metadata and the traffic of high dimensions of genuine file contents are sidetracked to find a way around the master to attain elevated performance and scalability. GFS takes a violent advance to offer fault tolerance, in which three reproduction of data are accumulated by default. GFS is personalized to rally the fussy burden for Google's data handing out and is not a regular file system.

Hadoop is intended to unfailingly store up very huge files across machines in a bulky cluster. It is a circulated parallel fault lenient file system. It is stimulated by the Google FS. Hadoop stores all folder and files as a series of blocks; all blocks in a file excluding the very last block are the similar size. Blocks fit in to a file are simulated for fault tolerance. The block size and duplication feature are systematize per file. Files are engraving once and contain rigorously one author at any instance. Hadoop is a top-level Apache

Corresponding Author: *: N.Durga*

assignment being build and worn by an international society of contributor, written in the JDK programming kit.

In a circulated structure level if we make your mind up to position enthusiastic high performance machines which are actually expensive, faults or disruptions are not recurrent. So ancestor like Google determined to utilize product hardware which is omnipresent and very cost efficient , but to make use of such hardware they comprise to make a plan choice of indulgence faults/ disruptions as customary position and system should capable to make progress from such malfunction. Hadoop urbanized on alike plan choices to finger faults. So evaluate luster, pvfs which system take for established faults are occasional and needs manual intervention to promise sustained services on supplementary hand Hadoop turns out to be very hearty and fault tolerant decision. Hadoop ensures that few malfunctions in the system won't interrupt sustained service of data through repeated imitation and transfer of household tasks from disastrous machines to breathe machines in Hadoop ranch obviously. Though it's declare that GFS has Same potential since it's not accessible to other companionship those potential cannot be availed.

Introduction to Hadoop

Hadoop offer a distributed file system and a structure for the investigation and revolution of very large data sets with the Map Reduce pattern. A central feature of Hadoop is the separation of data and calculation across numerous (thousands) of machines, and run application process in parallel close up to the data.

The list out 1 exhibits the gears of hadoop. Hadoop is an Apache assignment; all workings are accessible through the Apache open source license. Yahoo has geared up and put in to 80% of the nucleus of Hadoop (HDFS and MapReduce). HBase was initially designed at Powerset, now a sector at Microsoft.

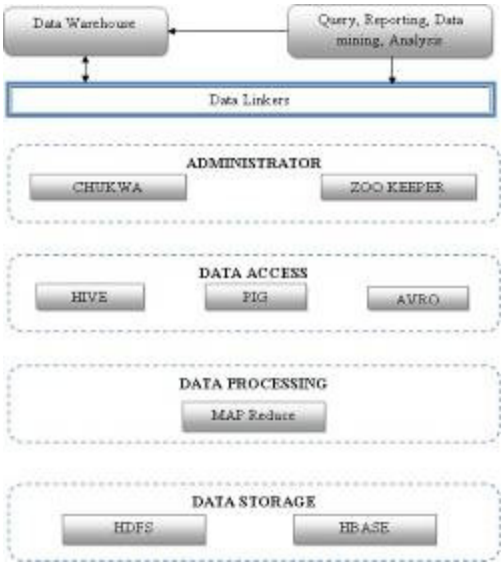


Fig 1: Hadoop System

HDFS	Distributed File System
Hbase	Column oriented table Service
PIG	Data Flow language and parallel execution framework
MapReduce	Distributed Computation FrameWork
Hive	data warehouse infrastructure
Chukwa	system for collecting management data
Avro	data serialization system
ZooKeeper	distributed coordinated sevice

List out 1.Hadoop project gears

Hive is initiated and developed at Facebook. Pig, Chukwa, and Zookeeper were initiated and developed at Yahoo! Avro was initiated at Yahoo! and co-developed with Cloudera. Hadoop Distributed File System is the file system section of Hadoop. While the crossing point to HDFS is decorative behind the UNIX file system, authenticity to principles was forfeit in favor of enhanced performance for the task at hand.

(HDFS) Hadoop Distributed File System

The (HDFS) Hadoop Distributed File System is a distributed file system intended to sprint on product hardware. It has various resemblances with accessible distributed file systems. Conversely, the dissimilarity from supplementary distributed file systems is momentous. HDFS is extremely fault-tolerant and is intended to be deployed on inexpensive hardware. HDFS supply tall throughput contact to program data and is appropriate for programs that have large data sets. HDFS was initially built as framework for the Apache web search engine development.

HDFS save file system metadata and program data independently. HDFS structural design consists of DataNode, NameNode, and Hadoop Distributed File System Client. A Hadoop Distributed File System Cluster may have thousands of DataNode and thousands of Hadoop Distributed File System clients per cluster, as every DataNode may perform multiple program tasks concomitantly. The most important characteristics of HDFS are that, it is extremely fault tolerance, appropriate for programs with large data sets. The below figure 2 exhibits the Hadoop Distributed File System structural design:

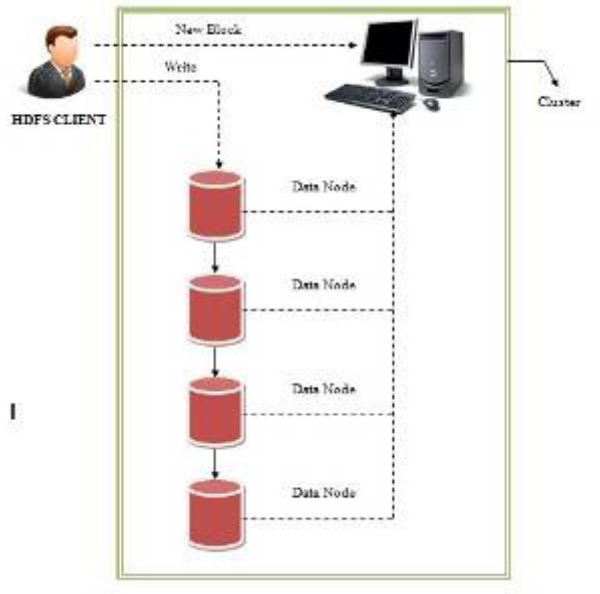


Fig 2: HDFS architecture

HDFS is master/slave design and consist of solo NameNode, a master server that administer the file system namespace and standardize admittance to files by consumers. ***Hadoop Distributed File System*** namespace is a hierarchy of files and folders. These files and folders which report characteristic like permissions, alteration, and access point in time namespace and disk space allocation.

A file is dividing into one or additional blocks and set of blocks are saved in DataNodes. A DataNodes saves data in the files in its home system and it does contain any awareness about HDFS file system. It saves each block of HDFS data in a partition file.

HDFS cluster has solo name nodes that administer the file system namespace. The current restriction that a cluster can hold only a solo name node outcome in the subsequent issues:

1. Scalability: Name node preserves the complete file system metadata in remembrance. The amount of the metadata is restricted by the physical memory accessible on the node. This ending in the subsequent issues:

a. **Scaling storage** – while storage can be sized by adding up additional data nodes/diskette to the data nodes, since additional storage fallout in additional metadata, the entirety storage file system can grip is restricted by the metadata size.

b. **Scaling the namespace** – the amount of files and folders that can be formed is restricted by the memory on name node.

To deal with these problem one promote bigger block volume, forming a smaller amount of larger files and via tools like the hadoop archives (har).

2. Isolation: No isolation for a multi-occupant atmosphere. An investigational client appliance that place high weight on the essential name node can bang a production Program.

3. Availability: While the propose does not put off constructing a failover mechanism, when a malfunction occurs the complete namespace and consequently the complete cluster is down.

A solo NameNode administer file system namespace, decide the mapping of file to blocks, and legalize admittance to files. In HDFS, all metadata is reserved in the remembrance of the solo NameNode, so it may turn into performance bottleneck as metadata numeral raises.

II. METADATA MANAGEMENT SCHEMES

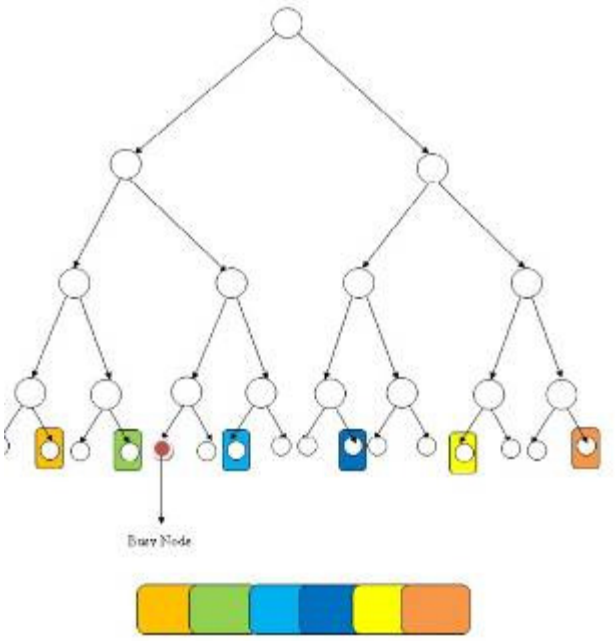
To allocate metadata surrounded by several servers some procedure are used like Tree separation, Hashing technique and reliable Hashing.

A. Tree separation

The **Tree** separation is worn in Coda file system and Ceph file system. The input design thought is that originally, the separation is carry out by hashing directories near the origin of the hierarchy, and when a server grow to be deeply overloaded, this demanding server robotically transfer some subdirectories to other servers with smaller amount loads. It also suggests prefix caching to resourcefully operate accessible RAM on all servers to promote advance the performance.

This loom has three foremost drawbacks. Initially, it supposes that there is a perfect load measurement proposal existing on each server and all servers from time to time swap over the load information. Second, when an MS link or unlink due to malfunction or healing, all directories require to be rehash to replicate the amend in the server transportation, which, in fact, produce a prohibitively high overhead in a PB-scale storage method. Third, when the hot spots of metadata process shift as the classification develop; recurrent metadata relocation in order to take away these hot spots may impress a large overhead and offset the reimbursement of load balancing.

In tree separation, namespace is separated into several directory sub trees, all of which is administer by single metadata servers. This tactic provides a good position because metadata in the identical sub tree is allocate to the identical metadata server, but metadata may not be consistently circulated, and the computing and relocate of metadata may produce a high instant and system overhead.



Busy Node Hashed across many MDS

Fig 3: Tree Separation

B. Hashing Technique

Hashing technique is worn in Lustre, zFs file system. Hashing technique utilize a hash utility on the path name to obtain metadata position. In this method, metadata can be circulated unvaryingly between clusters, but the directory vicinity characteristic is lost, and if the corridor is renamed, some metadata have to transfer. However, a severe trouble arises when an top folder is renamed or the whole amount of MSs vary the hashing mapping wants to be re-implemented, and this involve all exaggerated metadata to be transfer among MSs. even though the amount of the metadata of a file is little, a large amount of files may be involved. In picky, the metadata of all records has to be reposition if an MS links or unlinks. This may possibly direct to both disk and network traffic rush forward and origin severe performance filth. The hashing-based mapping advance can equilibrium metadata workloads and essentially has rapid metadata hunt for procedure, but it has sluggish directory process such as catalog the directory contents and renaming directories; in toting up, when the total quantity of MSs vary, rehashing all accessible files generates a unaffordable migration overhead.

As the over reveal two techniques has the focal drawback that metadata are not uniformly circulated in the system, to conquer these dependable hashing technique is worn.

C. Reliable Hashing

Reliable **hashing** is projected hash technique worn in Amazon. In essential Reliable hashing, the production variety of the hash utility is treated as a ring. Not merely the data is hashed, but also every node is hashed to a worth in the ring. Every node is accountable for the data in the variety among it and its precursor node. In Reliable hashing, the toting up and elimination of a node merely influence its neighbor nodes. An optimization of Reliable hashing is the opening of "virtual node". As an alternative of mapping a substantial node to a solitary spot in the ring, each material node is allocated to multiple situations, each of which is called an essential node. Through virtual node, workload and data is disseminated over nodes more unvaryingly.

A solitary NameNode administer file system namespace, establish the mapping of file to blocks, and legalize admittance to files. In HDFS, all metadata is reserved in the remembrance of the lone NameNode, so it may happen to performance bottleneck as metadata numeral increases. So in HDFS, we distorted the solitary NameNode design to multiple NameNodes, and the creator has projected a metadata management proposal.

1) System Design Architecture

The Figure 4 shows the structural design of metadata management method. The method mainly consists of four gears: Client, several NameNodes, NameNode administrator and Database. Client representation border to admittance metadata. To advance structure performance, some freshly worn metadata will be cached in consumer. NameNode is accountable for administrating metadata and commerce with metadata demand from Client.

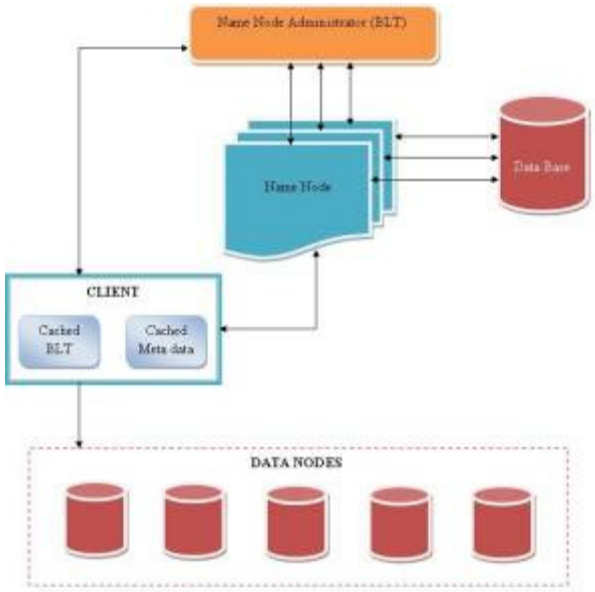


Figure 4: Metadata administration structure

The metadata in this manuscript consists of folder metadata and file metadata. Folder metadata consist of a hierarchical namespaces and directory characteristic, and file metadata consist of file characteristic and the mapping from file to statistics blocks. In general, a metadata is a ROW as

below:

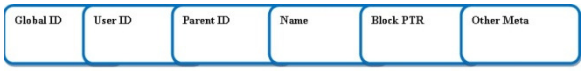


Fig 5: Metadata format

GLOBAL_ID is a universal sole identifier that is constant once the path is produced. USER_ID is an identifier of client that formed the lane. PARENTGLOBALID is the GLOBALID of the root folder of the path. OTHER_META keep other information, such as authorization, last admission time and revise time. BLOCK_PTR is the indicator to the file statistics blocks. The simplified metadata in NameNode is persevering into record sporadically. NameNode administrator provides a path for Client to get the objective NameNode. Besides, it administers the metadata allotment and load harmonizing among NameNodes by sporadically getting heartbeat communication from NameNodes.

2) *Metadata Partitioning*

Reliable hashing ring is divided in Q uniformly sized element and everyone is called "bucket" .Metadata is separation by buckets and equally circulated across NameNodes. Mapping of path's starting metadata to bucket is similar to Reliable hashing primary hash USERID and PARENT_GLOBAL_ID of the trail to give way its place p. march clockwise to locate 1st bucket with place larger than p.

3) *Metadata Access*

To systematize namespace hierarchy they have approve hash table. The figure 5 exhibits the NameNode data arrangement. For paradigm we desire to admittance to path /A/B/C/filename

- ✓ Client acquire userid and globalid of trail as ParentGlobalID
- ✓ Computes the Reliable hashing result
- ✓ Then client observe the cached BLT to locate out bucketid and NameNode I
- ✓ Sends the application to NameNode I in outline of <bucketid, userid, parentglobalid, filename> then that NameNode I distinguish its bucket array by bucketid.

In the manuscript we have also proposed the elucidation to metadata protection in memory. The key is "Log Replication". As the metadata is sporadically continue into database, they have got the most recent metadata by pertain the latest log proceedings on the last-persisted metadata in record. So they encompass just simulated the hottest log records relatively than memorial metadata. To evade log records agreement troubles linking primary and secondary's algorithm of Paxos is worn.

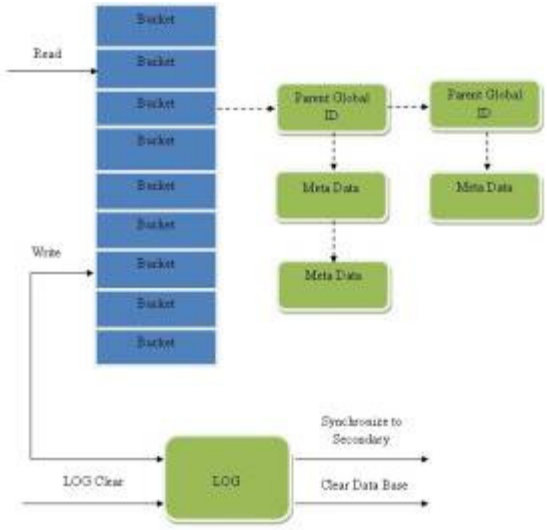


Fig 6: NameNode Data formation

Malfunction recognition is put into practice by heartbeat mechanism. Failure treatments comprise metadata healing, bucket re-contribution. In upturn all metadata supervise by the failed primary is improved. After the metadata in failed main has been recovered, NameNode administrator will donate the pail to other NameNode by amend BLT.

A NameNode can be further to or unconcerned from the cluster lacking system restart, and the metadata can be reallocated to remain load balance.

III. EVALUATION OF METADATA MANAGEMENT PROCEDURE

As scalability is the major problem in HDFS and solo NameNode has to function all metadata procedure; the complete weight is on one solo NameNode that can change the effectiveness, consistency and scalability of the organization. Several NameNode are commencing in the design to give out the load, but metadata allocation evenly among several NameNode is vital.

Reliable hashing technique when evaluate with other two techniques we got the answer as shown in below table:

List out 2: evaluation of Techniques

	Reliable Hashing	Hashing	Tree Separation
Performance	High	Low	Medium
Load Balancing	Yes	Yes	No
Reliability	High	Low	Low
Scalability	Highly Scalable	Moderately Scalable	Moderately Scalable

In Reliable hashing technique the routine is high as evaluate to other two techniques; since the allocation of

metadata and steering of metadata demand is successfully done via Reliable hashing which leads to enhancement in system concert. In this method the statistics bucket is separated into equal size which is additional evenly circulated over the NameNodes which leads to professional load balancing because of even allotment. Log scheme is used for storing and pre fetching of metadata. As the logs are saved into bucket look up table which is saved at client cache and equally simulated to NameNodes remembrance as well. Due to log duplication fault-tolerance is improved which causes high consistency. Reliable hashing consist the supplement of metadata or elimination of the same lacking distressing the cluster and it also reorganize the load which leads to correct load balancing which amplify scalability.

In Hashing technique the performance is small since hashing demolishes the neighborhood of metadata which causes the occasion to pre fetching and saving the metadata in buckets. The hash-based mapping advance can equilibrium metadata workloads and inherently has fast metadata search for procedure, but it has sluggish directory operations such as catalog the directory filling and renaming directories. In toting up, when the total amount of MSs Transform, rehashing all accessible files produce an unaffordable relocation overhead. In hashing the hash utility utilize the search-key of the pail. This search key is exceptional. Due to the matchlessness of search key in hashing reliance is generated which guide to short dependability.

In Tree separation the performance is intermediate evaluate to hashing technique. As tree separation uses N data structure in which the addiction is fashioned over core node and on parent node. Thus dependability decreases. We can say that evaluate to two techniques Reliable hashing is enhanced technique.

IV. CONCLUSION

We have seen the gears of hadoop and the HDFS in epigrammatic. As evaluate to other file system HDFS is an extremely fault tolerance system. The major drawback of HDFS was its solo NameNode which grip all metadata procedure. In this the disadvantage is conquer by commence multiple NameNodes in the arrangement. The additional problem arises is to handle metadata procedure between several metadata servers. In this manuscript we have evaluate three techniques Tree separation, hashing technique and Reliable hashing. To allocate the metadata consistently among the metadata server they have espouse Reliable hashing technique. As measure up to to other techniques Reliable hashing uniformly distributes weight to the server and use less memory. BLT makes available a proficient metadata repossession mechanism for Client to find the objective NameNode. To promise metadata accessibility under cluster malfunction, log duplication technology with Paxos algorithm is implemented. In addition, system performance profit from metadata caching and pre fetching.

REFERENCES

[1] Zhanye Wang ; Dept. of Comput. Sci. & Technol., Tsinghua Univ., Beijing, China ; Dongsheng Wang "NCluster: Using Multiple Active Name Nodes to Achieve High Availability for HDFS" Published in: High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on Date of Conference: 13-15 Nov. 2013 Page(s): 2291 – 2297.

[2] Azzedin, F. ; Inf. & Comput. Sci. Dept., King Fahd Univ. of Pet. & Miner., Dhahran, Saudi Arabia "Towards a scalable HDFS architecture" Published in: Collaboration Technologies and Systems (CTS), 2013 International Conference on Date of Conference: 20-24 May 2013 Page(s): 155 – 161.

[3] Dutta, R. ; Dept. of Comput. Sci. & Eng., Nat. Inst. of Technol. Karnataka, Surathkal, India ; Annappa, B. "A Scalable Cloud Platform using Matlab Distributed Computing Server Integrated with HDFS" Published in: Cloud and Services Computing (ISCOS), 2012 International Symposium on Date of Conference: 17-18 Dec. 2012 Page(s): 141– 145.

[4] Cheng Li-xin ; Suzhou Inst. of Nano-Tech & Nano-Bionics (SINANO), Suzhou, China ; Yue Yang ; Liu Yun-yun ; Ka Lok Man "A research on the storage structure and reconstruction algorithm of Meta data during the startup process of solid state storage systems" Published in: SoC Design Conference (ISOCC), 2013 International Date of Conference: 17-19 Nov. 2013 Page(s): 263 – 266.

[5] Zhen-Chun Huang ; Dept. of Comput. Sci. & Eng., Tsinghua Univ., Beijing ; Guo-qing Li "Meta-Data Adapter: A Way to Enable Spatial Information Data Grid" Published in: Future Generation Communication and Networking Symposia, 2008. FGCNS '08. Second International Conference on (Volume:5) Date of Conference: 13-15 Dec. 2008 Page(s): 87 – 92.

[6] Narayan, S. ; InfoBlox Inc., Santa Clara, CA, USA ; Bailey, S. ; Daga, A. "Hadoop Acceleration in an OpenFlow-Based Cluster" Published in: High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion: Date of Conference: 10-16 Nov. 2012 Page(s): 535 – 538.

[7] Mandal, A. ; Renaissance Comput. Inst., Univ. of North Carolina at Chapel Hill, Chapel Hill, NC, USA ; Yufeng Xin ; Baldine, I. ; Ruth, P. "Provisioning and Evaluating Multi-domain Networked Clouds for Hadoop-based Applications" Published in: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on Date of Conference: Nov. 29 2011-Dec. 1 2011 Page(s): 690 – 697.

[8] Xiaoyi Lu ; Islam, N.S. ; Wasi-Ur-Rahman, M. ; Jose, J. "High-Performance Design of Hadoop RPC with RDMA over InfiniBand" Published in: Parallel Processing (ICPP), 2013 42nd International Conference on Date of Conference: 1-4 Oct. 2013 Page(s): 641 – 650.

